

IN THE DRAWINGS

Please amend Figure 1 as shown in the enclosed replacement sheet. Figure 1 has been amended to include reference signs. No new matter has been added.

REMARKS

Please reconsider the present application in view of the above amendments and the following remarks. Applicant thanks the Examiner for carefully considering the present application. Applicant further thanks the Examiner for the courtesies extended in the telephonic Examiner Interview of May 18, 2006.

I. Disposition of the Claims

Claims 2-12 and 21 were pending. By way of this reply, claims 2-12 and 21 have been canceled without prejudice or disclaimer, and new claims 22-39 have been added. No new matter has been added by way of these amendments. Accordingly, claims 22-39 are currently pending.

II. Substitute Specification

As discussed in the telephonic Examiner Interview of May 18, 2006, a Substitute Specification has been prepared for purposes of clarification only. The clean and marked-up versions of the Substitute Specification are enclosed with this Reply. No new matter has been added. Accordingly, entry of the Substitute Specification is respectfully requested.

III. Rejection(s) under 35 U.S.C. § 112

§ 112, 1st

Claims 2 and 21 were rejected under 35 U.S.C. § 112, first paragraph, as failing to meet written description and enablement requirements. However, by way of this reply, claims 2 and 21 have been canceled, and thus, the rejections of these claims are now moot. To the extent that

these rejections are believed to apply to new independent claims 22, 34, or 37, these rejections are respectfully traversed.

New independent claims 22, 34, and 37 recite that a second portion of data for a web page is received after sending a transformed first portion of the web page data to a wireless computing device. As shown, for example, in the “Bluelark” system in Figure 1, a proxy server “streams content” on to a wireless computing device by transforming data “on the fly” (such a technique being referred to as “Rapid Streaming” technology on page 2). This streaming/transformation mechanism is in contrast to that used by “other systems,” in which a proxy server “*waits for [an] entire page*” (emphasis added) to be received before sending web page data to the wireless computing device. It is clear from Figure 1 that Applicant contemplated and enabled a proxy server receiving portions of web page data after sending other portions of the web page data to a wireless computing device. Thus, the limitations of new independent claims 22, 34, and 37 that are similar to those in now-canceled claims 2 and 21 meet the written description and enablement requirements of § 112, first paragraph. Accordingly, withdrawal of the § 112, first paragraph, rejections is respectfully requested.

§ 112, 2nd ¶

Claims 3-10 were rejected under 35 U.S.C. § 112, second paragraph, as failing to particularly point out and distinctly claim the subject matter which the Applicant regards as his/her invention. However, by way of this reply, claims 3-10 have been canceled, and thus, the rejections of these claims are now moot. Accordingly, withdrawal of the § 112, second paragraph, rejections is respectfully requested.

IV. Rejection(s) under 35 U.S.C. § 102

Claims 2-12 and 21 were rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,430,624 (“Jamtgaard”). For the reasons set forth below, these rejections are respectfully traversed.

“A claim is anticipated only if *each and every element* as set forth in the claim is found, either expressly or inherently described, in a single prior art reference.” *See Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631 (emphasis added); MPEP § 2131. New independent claims 22, 34, and 37 in part require receiving a second portion of web page data *after sending* a [transformed] first portion of the web page data to a wireless computing device. Jamtgaard fails at least to disclose this limitation of the claimed invention.

Jamtgaard discloses a system in which different types of content may be delivered to different appliances having different browser specifications. *See Jamtgaard*, Abstract. As described in detail:

[T]he method for content delivery may include intelligently harvesting content from a web page to provide that content to a plurality of different information appliances having different screen sizes. The intelligent harvesting may convert the content into a proprietary relational markup language (RML) and generate a tree and then a document object model from the RML content. The tree may then be analyzed and searched using a set of processing rules in order to generate content screens customized to each information appliance. A typical card builder may build the card corresponding to the customized content and a typical deck builder may build a deck of cards corresponding to the one or more display screens that make up the content for the particular information appliance. The deck of cards may then be converted into a presentation format and protocol for the particular information appliance and sent to that information appliance.

See Jamtgaard, col. 2, line 60 – col. 3, line 9. Referring to Figure 2, the server **12** that performs the features of the cited passage “receives incoming content on-the-fly from an Internet content

provider 13[,] thereby allowing for dynamic information generation.” See Jamtgaard, col. 4, lines 24 – 26. However, as discussed during the Examiner Interview of May 18, 2006, although the server 12 may receive web page data “on-the-fly,” web page data for a particular wireless device is not sent to that wireless device until a “presentation shoe” has been built in the server 12. In other words, only after the server 12 has received all the data for a particular web page and converted that data for a particular wireless device (i.e., “once the presentation shoe is completed by the layout engine”), will the server 12 send the content to the appropriate wireless device(s). See Jamtgaard, col. 9, lines 40 – 47. Thus, in Jamtgaard, transformed content for a web page is sent to a wireless computing device only after all the web page data has been received by the proxy server. Accordingly, Jamtgaard fails to disclose, either inherently or explicitly, the proxy server receiving a second portion of data for a web page *after sending* a first portion of the web page data to a wireless computing device as required by the claimed invention.

In view of the above, Jamtgaard fails to disclose the invention as recited in independent claims 22, 34, and 37. Thus, independent claims 22, 34, and 37 are patentable over Jamtgaard. Dependent claims 23-33, 35, 36, 38, and 39 are allowable for at least the same reasons. Accordingly, withdrawal of the § 102 rejections is respectfully requested.

V. Conclusion

The Examiner is encouraged to contact the undersigned attorney if it would be beneficial to further advance the prosecution of the application.

Please apply any charges not covered, or any credits, to Deposit Account 19-2555 (Reference No. 24772-05944).

Date: 06/05/06

Respectfully Submitted,
Jack B. Strong, et al.
By: W. H. Qureshi

Wasif H. Qureshi, Attorney of Record
Registration No. 51,048
FENWICK & WEST LLP
801 California Street
Mountain View, CA 94041
Phone: (650) 335-7121
Fax: (650) 938-5200



SUBSTITUTE SPECIFICATION – “MARKED-UP” VERSION

APPARATUS AND METHOD FOR SUPPLYING ELECTRONIC CONTENT TO NETWORK APPLIANCES

Inventors: Jack B. Strong, John N. Lehner, Jonathan J. Kleid, and Vivek Patel

CROSS REFERENCE TO RELATED APPLICATIONS

[0001] This application is based on and claims priority under 35 USC § 119(e) of U.S. Provisional Patent Application No. 60/212,147, filed on June 16, 2000 and entitled “Apparatus and Method for Supplying Electronic Content to Network Appliances”, which is incorporated by reference herein.

COPYRIGHT NOTICE

[0002] A portion of the disclosure of this patent document contains material which is subject to copyright protection. The copyright owner has no objection to the facsimile reproduction of the patent document or the patent disclosure, as it appears in the Patent and Trademark Office patent file or records, but otherwise reserves all copyright rights whatsoever.

INVENTORS

Jack B. Strong, John N. Lehner, Jonathan J. Kleid, and Vivek Patel

~~CROSS REFERENCE TO RELATED APPLICATIONS~~

~~This application claims priority under 35 USC § 119(e) from Provisional~~

~~Application Serial No. 60/212,147, entitled "Apparatus and Method for Supplying Electronic Content to Network Appliances," filed on June 16, 2000, which is incorporated by reference herein.~~

BACKGROUND

[0003] Until recently, the vast majority of internet users accessed the World Wide Web via personal computers and workstations. Web designers could safely assume that the devices accessing their sites would have similar display capabilities, processor speed, and connection bandwidth. With careful design, it was possible to produce a single version of a web site that displayed satisfactorily in nearly all browsers. Unfortunately, this is no longer possible. The number and variety of so-called "network appliances" has exploded. Millions of internet users now log on via set top boxes, personal digital assistants (PDAs), and cellular telephones. Analysts have predicted that more people will soon access the web via network appliances than via personal computers. Now, web designers must take into account that browsers accessing their sites may display just a few words of text at a time, or they may display hundreds. Other browsers have no display at all and instead read the content to the user through a telephone. Processor speeds and bandwidth may vary by an order of magnitude. To make matters worse, different devices support different sets of file formats, often with no intersection at all.

[0004] Web content providers that need to support users on all these devices typically have to provide multiple versions of their sites. Amazon.com currently

has three versions of its site: the conventional HTML version; an HDML version for display on first generation cell phone browsers; and a “web clipping” version for certain handheld computing devices. A fourth version will be for WML capable cell phones. Only content providers with the deepest pockets can afford to author so many different versions of their sites. Even for giants like Amazon.com, this approach is clearly not scalable as the variety of network appliances continues to increase.

[0005] As with many problems in computer science, this dilemma can be at least partially solved by adding a level of indirection. In internet terms, the level of indirection between a browser and a web server is called a proxy server. The model is simple. When a browser needs to retrieve a document from the web, it sends a request to the proxy server using the browser's native protocol. The proxy then retrieves the document from the server. If the document happens to be in a format that the browser is capable of displaying, the proxy will simply forward it along. This is where the process ends for most, if not all, traditional proxy servers.

SUMMARY

[0006] Embodiments of the present invention relate to a system and method for communicating data from a web content provider to a wireless computing device (e.g., a PDA, a cellular phone) via a proxy server. The proxy server transforms web content “on-the-fly” into a streamlined format, optimizes it for display, and

delivers it to the device, where it is progressively rendered by a browser installed on the device.

[0007] The features and advantages described herein are not all inclusive, and, in particular, many additional features and advantages will be apparent to those skilled in the art in view of the following description. Moreover, it should be noted that the language used herein has been principally selected for readability and instructional purposes and may not have been selected to circumscribe the present invention.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Fig. 1 is a simplified diagrammatic representation of an embodiment of the invention.

[0009] Fig. 2 is a simplified illustration of a progressive rendering technique employed in an embodiment of the invention.

[0010] Each of the figures referenced above depict an embodiment of the present invention for purposes of illustration only. Those skilled in the art will readily recognize from the following description that one or more other embodiments of the structures, methods, and systems illustrated herein may be used without departing from the principles of the present invention.

~~EXECUTIVE SUMMARY~~

1. Executive Summary

Bluelark's Technology Transforms the Web for PDAs

Bluelark Systems has developed a "Rapid Transformation Server" and a browser that combine to give users of wireless PDAs fast, easy access to all types of web pages for the first time. The server transforms web content into a streamlined format and optimizes it for display on a small screen, then delivers it to the PDA where it is rendered by Blueark's "Blazer" browser. This allows users to view content created in any major format (including HTML, WML, Web Clipping and packaged "Channels") through a single browser, instead of having to use separate applications for each one as at present. Web pages load ~~much~~ faster on Blazer than on any other PDA browser because Blueark's proprietary content streaming technology eliminates the bottlenecks found in other systems.

~~We Will Freely Distribute the Browser to End-Users, ...~~

~~We will aggressively promote free downloads of Blazer to stimulate usage. We expect to gain rapid adoption and "mind share" because there is a widespread need for mobile web access on PDAs, and existing products have serious deficiencies.~~

~~... Sell the Technology to ISPs and Portals, ...~~

~~We will sell the server-based transformation technology to Wireless Internet Service Providers (WISPs) and wireless portals, along with the right to distribute OEM versions of Blazer that are configured to use the customers' own proxy server and start page. Reactions by these companies to pre-release versions of Blueark's technology have been unanimously positive. The transformation process also works well with Microsoft's Pocket IE, and we will extend it to other PDA browsers as necessary.~~

~~... And License the Platform for Custom Applications~~

~~Blueark's technology is the ideal platform for thousands of custom wireless PDA applications, ranging from Sales Force Automation to Electronic Banking. We will license both the server and the browser components to developers in each vertical market.~~

~~No Competitor has Comparable Technology~~

~~Blueark has a unique combination of technologies, including:~~

- ~~• On the fly HTML to WML transcoding~~
- ~~• On the fly content transformation for optimal display on a PDA screen~~
- ~~• Proprietary "Rapid Streaming" technology, which delivers content in a fraction of the time taken by other PDA browsing systems~~
- ~~• A Palm OS browser that renders a superset of WML, with Blueark specific extensions, to improve the browsing experience~~

~~Various other companies have fragments of this technology chain (e.g. AvantGo and PumaTech both have HTML browsers for the Palm OS), but Blueark has a 6-12 month lead in delivering a complete, viable WML/WML browsing solution for PDAs.~~

~~We Require Funding to Grow our Team and our Infrastructure~~

~~Development to date has been funded by the founders. Blueark needs additional capital to hire sales, engineering and customer support personnel, to expand our server side infrastructure and to purchase encryption technology for secure commercial transactions.~~

~~Blueark has Strong Technical and Management Experience in this Field~~

~~Blueark has unrivalled experience in delivering information services to mobile computers. Blueark's core team collaborated among other projects on the design and development of the qRx software for ePocrates Inc., which is by far the dominant healthcare application for PDAs with 60,000 physician users as of May 2000. Blueark has 7 employees as of 6/5/00: 5 software engineers, 1 business development manager and 1 UI/graphic design specialist. We are actively recruiting to grow our engineering and business development teams.~~

~~SERVER SIDE~~

Project Overview

The goal of this project is to produce an HTTP proxy server with support for modules that can perform arbitrary transformations on requested files before forwarding them to the user agent. The base server will include modules for converting among several popular document markup languages and reducing the file size of common web image formats. In this document, we will examine the market opportunity and assess the key issues that will affect the outcome of the project.

Opportunity Statement

Until recently, the vast majority of internet users accessed the World Wide Web via personal computers and workstations. Web designers could safely assume that the devices accessing their sites would have similar display capabilities, processor speed, and connection bandwidth. With careful design, it was possible to produce a single version of a web site that displayed satisfactorily in nearly all browsers. Unfortunately, this is no longer possible. The number and variety of so-called "network appliances" has exploded. Millions of internet users now log on via set top boxes, personal digital assistants, and cellular telephones. Analysts predict that by the year 2002, more people will access the web via network appliances than via personal computers. Now, web designers must take into account that browsers accessing their sites may display just a few words of text at a time, or they may display hundreds. Other browsers have no display at all and instead read the content to the user through a telephone. Processor speeds and bandwidth may vary by an order of magnitude. To make matters worse, different devices support different sets of file formats, often with no intersection at all. Web content providers that need to support users on all these devices have no choice but to provide multiple versions of their sites. Amazon.com currently has three versions of its site: the

conventional HTML version, an HDML version for display on first generation cell phone browsers, and a Web Clipping version for the PalmVII. A fourth version is under development for WML capable cell phones. Only content providers with the deepest pockets can afford to author so many different versions of their sites. Even for giants like Amazon.com, this approach is clearly not scalable as the variety of network appliances continues to increase.

As with many problems in computer science, this dilemma can be at least partially solved by adding a level of indirection. In internet terms, the level of indirection between a browser and a web server is called a proxy server. The model is simple. When a browser needs to retrieve a document from the web, it sends a request to the proxy server using the browser's native protocol. The proxy then retrieves the document from the server. If the document happens to be in a format that the browser is capable of displaying, the proxy will simply forward it along. This is where the process ends for most proxy servers today.

Room for Improvement

We propose to add a critical new feature. If the document is not in a format that the browser is capable of displaying, the server will attempt to transform it into an acceptable format. For example, if the server sends a document marked up in HTML, and the browser only supports WML, the proxy can run the document through an HTML to WML translation module. This model can be extended beyond simple file format translations to encompass display optimizations as well. For example, if the server sends a 256 color GIF image, and the device supports only 4 shades of gray, then the server can reduce the bit depth of the image, thereby saving the device the processing overhead of doing the conversion itself and reducing the file size to optimize transmission speed over a low bandwidth connection. The entire conversion process is transparent to both the web content developer and the browser user.

State of the Art

AOL developed the first non-academic transforming proxy server to use with the service's proprietary web browser in 1995. The primary purpose of the server was to save precious network bandwidth by stripping comments and unsupported markup from HTML documents and converting GIF's and JPEG's to a highly compressed format before forwarding them to the client. Other proprietary browser manufacturers followed suit, developing proxy servers to compress, and in some cases reformat, web content. WebTV developed such a server for its set top box. AvantGo and ProxiNet developed similar servers for their PDA-based browsers. The most flexible proxy server of this genre is Spyglass's Prism, which supports their embedded browser on a variety of platforms.

Other vendors have taken the opposite approach, developing web server plug-ins which allow content providers to create multiple versions of their web site more easily. IBM's WebSphere Transcoding Publisher is a prime example. OnlineAnywhere offered a similar product until they were acquired by Yahoo in February. To our knowledge, there are no commercial grade generic HTTP proxy servers which support content transformation.

Competitive Advantage

Our proxy frees end users (both content producers and content consumers) from concerning themselves with the proxy. The products mentioned above all require either the content provider or browser developer to maintain the proxy. This includes not only maintaining a server farm, but also keeping the transformation software up to date on all the latest file formats and device capabilities. By dissociating the transformation engine from both the server and client, our project will allow a third party application service provider to take over the task of maintaining the proxy. The software can be upgraded to support new devices and formats transparently to both the content provider and the end user. We believe that this degree of transparency is crucial to

widespread adoption of proxy-based transformation technology.

Additionally, our team has considerable experience developing scalable network servers and in transforming content for devices with limited display capabilities. We recently applied this experience to develop a drug database for the Palm which automatically updates itself via the internet. The application is currently in active use by over 30,000 physicians.

Other features

Dynamic construction and configuration of document transformation streams based on user-agent profiles stored locally on the proxy server, provided by the user agent, or retrieved from another server based on information provided by the user agent. The profiles specify supported content types, display characteristics of the device, and transformation preferences, and allow the server to optimize display for user agents whose capabilities are not known at the time the server is deployed.

Conclusion

Though there is significant risk in this project, there are also significant potential rewards. The web content transformation market is far from mature, and there is still no clear market leader. A superior product could significantly grow the market as well as capture existing market share. Even if it is not possible to generically transform content targeted for PC browsers to a form suitable for network appliances, a product which does a good job of converting among the various formats intended for devices with similar capabilities would be still be useful. For example, there are currently five formats for wireless devices: HDML, WML, PQA (also know as the Web Clipping format), XHTML Basic, and AvantGo's non-standard HTML subset. A proxy server which supports converting to and from these five formats would certainly find customers.

CLIENT SIDE

Design Document Voyager WAP Browser for the PalmOS

Architecture

The application will be structured as an object-oriented Palm program. One set of classes will be primarily focused on aspects of the Palm environment, such as responding to launch codes and user events. A second set of classes will be responsible for browser-specific tasks, such as creating a parse tree from a WML document and displaying the data. A third set will be responsible for interfacing between OS-independent browsing code and the PalmOS. And finally a fourth will be comprised of generic, reusable components such as vectors and iterators. Sub-goals of this project are to create a reusable framework for Palm application development, as well as a platform-independent browser kernel. We will not sacrifice the project to achieve these goals, but we will keep them in mind during design and implementation, because these sub-goals naturally lead to clean abstractions.

Palm-Specific Components

Application

- Top-most level of Palm application
- Determines whether the current operating environment is suitable for this application (i.e. make sure the OS version number is not too old)
- Handles Palm launch codes (e.g. normal launch, soft reset, beam received).
- Receives all user input; must dispatch commands to the appropriate handlers.
- An application is defined by a series of forms, so this class must manage a set of forms that are each active at different times.

Form

- Represents abstract form (the Palm equivalent to a window). Meant to be subclassed by forms with specific functions.
- Supplies UI functions common to all (or at least most) forms.
- Contains zero or more views, arranged in some form-specific way.

View

- — Represents abstract region for displaying some sort of content or UI.
- — Not nestable (i.e. a view cannot contain another view)
- — Gives forms more flexibility since views are location independent (i.e. all drawing within a view is relative to the view's coordinates)

Prefs

- — Manages the storage of persistent preferences used by the application.

EventHandler

- — Abstract interface for any component that can handle user events (i.e. Form, View)

Browser Specific Components

Expat

- — Collection of open source C files which parse XML files.
- — We will use expat as a shared library to perform parsing, and will subsequently build a parse tree based on its output.

XMLParser

- — Wrapper around expat and the parse tree (expat could be replaced by a different parser, and only this class would be affected)
- — Manages the construction of the tree, using expat to build a tree composed of XMLElements and XMLAttributes.

XMLElement

- — Single XML tag, with attributes and data

XMLAttribute

- — Represents a single XML attribute

XMLGlyph

- — Graphical representation of XMLElement.
- — Based on recursive composition technique in *Design Patterns*, Chapter 1 (by the Gang of

Four):

- A glyph can be a container for other glyphs, or can be a leaf.
- Glyphs draw themselves, and handle UI tasks within its boundaries.
- Examples of glyphs are images, blocks of text, and lines of text.

HTTPProtocol

- Implements HTTP Protocol for any underlying session mechanism (i.e. TCP)
- Possibly composed of public domain source code.

PalmOS/WML Browser Glue Components

WMLForm

- Subclass of Form, defines browser specific form features, such as back and forward arrows.
- Contains a WMLView

WMLView

- Displays WML content. Can be embedded in any application that implements the Form and View classes described above.
- Gives WMLGlyph a display area for layout
- Responsible for scrolling WML content

WMLSession

- Responsible for retrieving data, and supplying pages on demand to the WML browser.
- The underlying mechanisms will be pluggable, meaning regardless of whether a modem is used, or a Bluetooth module communicating to a cell phone, the rest of the browser will work in the same way.

TCPSession

- Subclass of WMLSession which uses Palm's built-in TCP stack
- Provides socket level instructions to enable HTTP

Generic Classes

Vector

- ~~Easy to use data structure for managing groups of data~~

Iterator

- ~~Iterator for vector, or any other class which represents a collection of objects~~

Types

- ~~Defines commonly used types such as Point and Rectangle~~

Component Interface Design

Application

- ~~*main*~~: entry point for application. Calls appropriate function depending on the launch code passed in. Performs suitability checks on environment.
- ~~*eventLoop*~~: dispatches commands to appropriate eventHandlers, which have registered themselves. Keeps track of current handler.
- ~~*registerHandler*~~: called by forms that can handle events, to tell the application that they should be used with forms of certain ID's. This is necessary because in the PalmOS, forms are labeled with ID numbers. By registering themselves, form objects become associated with the Palm's notion of a form.
- ~~*launchNormal*~~: virtual function called by main when program is launched normally. Can be subclassed by applications to customize behavior.

Form

- ~~*formLoad, formOpen, formClose*~~: Called by main event loop when this object is the current handler. *formLoad* should load resources needed to display the form, without writing anything to the screen. *formOpen* should draw the form itself. *formClose* should erase the form and release any resources.
- ~~*draw*~~: Called by eventLoop (and possibly other functions) when the form's contents should be drawn/redrawn.
- ~~*addView*~~: Adds new view to the form.
- ~~*handleEvent*~~: Processes form specific events

View

- ~~*handleEvent*~~: Processes view specific events
- ~~*getBounds*~~: Returns the area used by the view

Prefs

- ~~*read*~~: reads prefs from storage into temporary data structure

- ~~*write*: writes prefs from temporary data structure into storage~~
- ~~Subclasses should provide assessors/modifiers for specific preferences.~~

EventHandler

- ~~*doesHandleType*: Returns true if this object handles the specified event type (i.e. menu action)~~
- ~~*doesHandleObject*: Returns true if this object handles the specified object (i.e. form with ID #4300)~~
- ~~*handleEvent*: Actually handle the event. (Must be overridden by subclass)~~

Expat

- ~~*XML_SetElementHandler*: Used to set the callback for new elements. Callback is called for each XML tag seen.~~
- ~~*XML_SetCharacterDataHandler*: Used to set callback for data handler.~~

XMLParser

- ~~*parse*: parses specified document, returns root element of document (of type XML_Element)~~

XMLElement

- ~~*addAttribute*: adds attribute to element~~
- ~~*getAttribute*: returns vector of attributes~~
- ~~*getTag*: returns tag name~~

XMLAttribute

- ~~assessor/modifier for name, value pair~~

WMLGlyph

- ~~*intersects*: returns true if this glyph intersects specified point~~
- ~~*insert*: inserts a sub glyph into this glyph~~
- ~~*activate*: called when the Palm UI element containing the glyph is activated~~
- ~~*deactivate*: called when the Palm UI element containing the glyph is deactivated~~
- ~~*draw*: recursively draws glyph, and all its sub glyphs~~

- ~~*getExtent*: returns bounds of glyph~~
- ~~*parent*: returns parent glyph~~
- ~~*children*: returns vector of children glyphs, if any~~

HTTPProtocol

- ~~*sendRequest*: sends request to server, returns requested data. Uses underlying session protocol as provided on device.~~

WMLForm

- ~~overrides functions of Form, adding appropriate behavior~~

WMLView

- ~~overrides functions of View, adding appropriate behavior~~

WMLSession

- ~~*set View*: way for view to register itself with a session.~~
- ~~*goTo*: called by view to get data of passed in URL~~

TCPSession

- ~~*goTo*: overridden from WMLSession~~
- ~~provides underlying layer for HTTPProtocol to interact with, such as open socket, and read/write data.~~

Vector

- ~~*add*: adds element to end of vector~~
- ~~*insert*: inserts element into some position in vector~~
- ~~*remove*: removes specified element~~
- ~~*removeAll*: Empties vector~~
- ~~*getElementAt*: returns specified element~~
- ~~*getNumElements*: returns count of all elements.~~

Iterator

- *hasNext*: returns true if there are more elements
- *getNext*: returns next element

Data Structure Design

The most important data structures are the representations of the WML document. After parsing, a simple parse tree is created. The parse tree's structure is based on the notion that a child node is nested within their parent in the corresponding XML. For example, if a run of text looks like:

~~<p><i>text</i></p>~~

then the element representing 'i' will be a child of the element representing 'p'. Further, the text element will be a child of i. XML attributes are represented as a vector of string pairs, where each pair corresponds to a name and value.

~~This parse tree is independent from the glyph tree, although the two trees are similar in structure. The glyph tree has at least one glyph per parse tree element, and possibly more. For example, a block of text that spans multiple lines is represented as a single element in the parse tree. In the glyph tree, however, it will become multiple glyphs—one for each line, and one for the text block itself. This is based on the Composite pattern, described in *Design Patterns*.~~

~~Vectors are used extensively as a convenient abstraction on top of arrays. We would like to have used the STL for data structures such as this, but unfortunately it is not available on the Palm, as far as we know. We can take advantage of the situation and optimize our own data structures for use with the specific constraints we face.~~

Algorithm Design

~~The crux of the browser is the layout algorithm. It must be able to flow a wide variety of objects, and store their position in order to respond to user input (i.e. clicking on a link). It must be flexible enough to work concurrently with data download, so it must operate incrementally. It must be efficient, since computing resources are limited on the Palm. To facilitate these needs, we use the glyph tree to represent~~

on screen information, as described in the previous section. The glyph tree is flexible, in that a block of text can be reformatted into its constituent lines at any point and redrawn, if for example an image is newly downloaded and changes the flow of the document.

The glyph draw function will be implemented in a relatively coordinate independent way. This means that a glyph will be passed a display area in which to draw, which it can treat as its own canvas. Each glyph must also calculate the sub areas available to its sub glyphs. The rendering algorithm is then composed of all glyph types following this general contract. As long as glyphs draw only in the space allotted, and gives their sub glyphs adequate screen real estate, then the contract is fulfilled. Because of the recursive nature of the algorithm, considerable information can be cached. For example, the extent of a glyph can be stored, so the `getExtent` function will not need to examine all sub glyphs to recompute the extent.

DETAILED DESCRIPTION

[0011] In the following description of embodiments of the present invention, numerous specific details are set forth in order to provide a more thorough understanding of the present invention. However, it will be apparent to one skilled in the art that the present invention may be practiced without one or more of these specific details. In other instances, well-known features have not been described in detail to avoid unnecessarily complicating the description.

[0012] In general, embodiments of the present invention relate to a system and method for communicating data from a web content provider to a wireless computing device (e.g., a PDA, a cellular phone) via a proxy server. The proxy server transforms web content “on-the-fly” into a streamlined format, optimizes it for display, and delivers it to the device, where it is progressively rendered by a browser installed on the device.

[0013] When a document (e.g., a web page) received by the proxy server is not in a format capable of being displayed by a requesting wireless device, the proxy server will attempt to transform the document into an acceptable format. For example, if the web content provider sends a document marked up in HTML, and the browser of the requesting wireless device only supports WML, the proxy server can run the document through an HTML-to-WML translation module. Other formats possibly supported by wireless devices to which the proxy server may transform format include, for example, HDML, PQA (also known as the “web clipping” format), XHTML, Basic, and AvantGo’s non-standard HTML subset.

[0014] Further, in one or more embodiments, this model can be extended beyond

simply file format translations to encompass display optimizations as well. For example, if the web content provider sends a 256-color GIF image, and the requesting wireless device supports only 4 shades of gray, the proxy server may then reduce the bit depth of the image, thereby (i) saving the wireless device the processing overhead of doing the conversion itself and (ii) reducing the file size to optimize transmission speed over a low bandwidth connection.

[0015] Moreover, those skilled in the art will appreciate that the conversion process is transparent to both the web content developer and the user of browser of the requesting wireless device. In other words, a proxy server in accordance with one or more embodiments frees end users (both content producers and content consumers) from concerning themselves with the proxy server. Typical prior art proxy servers require either the content provider or browser developer to maintain the proxy server. This included not only maintaining a server farm, but also keeping the transformation software up-to-date on all of the latest file formats and device capabilities. In one or more embodiments, by disassociating the transformation engine from both the server and the client, a third party application service provider may maintain the proxy. Further, the transformation software may be upgraded to support new devices and formats in a manner that is transparent to both the content provider and the end user. This degree of transparency is believed to be important to the widespread adoption of proxy-based transformation technology. Moreover, in one or more embodiments, the transformation software may be updated automatically via, for example, the Internet.

[0016] The transformation of content by a proxy server in accordance with one or

more embodiments may be dynamic and/or based on user-agent profiles stored locally on the proxy server, provided by the user-agent, and/or retrieved from another server based in information provided by the user-agent. The profiles may specify supported content types, display characteristics of the device, and/or transformation preferences, and allow the proxy server to optimize a display for user-agents whose capabilities are not known at the time the proxy server is deployed.

[0017] Now referring to Figure 1, prior art proxy servers 12 (shown as “other systems”) wait for an entire web page to be received from a web content provider 10. Once the data for the entire web page has been received, the proxy server 12 transforms the data to a format for display by a web browser 14 of a requesting wireless device 20. The web browser 14 then waits for the entire web page data to be received prior to displaying the web page.

[0018] Still referring to Figure 1, in one or more embodiments, a proxy server 16 receives web page data in a streamlined manner and transforms received web page data “on-the-fly”. In other words, the proxy server 16 transforms and sends for display by a web browser 18 web page data as the proxy server 16 receives web page data from the web content provider 10. Thus, the proxy server 16 may receive portions of data for a web page after other portions of the web page data have already been transformed and sent by the proxy server 16. Moreover, the web browser 18 in accordance with one or more embodiments is arranged to progressively render received web page data. In other words, the web browser 18 may display the web page data as it receives it; it does not have to wait for all of the web page data to be received.

[0019] Thus, the proxy server 16 transforms web content into a streamlined format and optimizes it for display on a screen of a requesting wireless device 20. The requesting wireless device 20 may have a web browser 18 that is arranged to progressively render received web page data. In such a manner, users may view content created in any format through a single web browser (instead of having to use separate applications) (see Figure 2), and do so in an environment in which web pages load relatively quickly due to the content streamlining mechanism described with reference to Figure 1.

[0020] Accordingly, a system in accordance with one or more embodiments involves at least one of on-the-fly transformation, on-the-fly content transformation for optimal display on a device screen, "rapid streaming" technology, and a web browser that progressively renders received web page data.

[0021] A web browser in accordance with one or more embodiments will now be described. The web browser application may be structured as an object-oriented program. One set of classes may be primarily focused on aspects of a particular computing environment, such as responding to launch codes and user events. A second set of classes may be responsible for browser specific tasks, such as creating a parse tree from a WML document and displaying the data. A third set may be responsible for interfacing between OS independent browsing code and operating system code. A fourth set may include generic, reusable components such as vectors and iterators. Sub-goals may be to create a reusable framework for application development, as well as a platform independent browser kernel. Those skilled in the art will note that these sub-goals, among

others, may naturally lead to clean abstractions.

Application

- Top-most level of application
- Determines whether the current operating environment is suitable for this application (i.e. make sure the OS version number is not too old).
- Handles launch codes (e.g. normal launch, soft reset, beam received).
- Receives all user input; must dispatch commands to the appropriate handlers.
- An application is defined by a series of forms, so this class must manage a set of forms that are each active at different times.

Form

- Represents abstract form (e.g., a window). Meant to be sub-classed by forms with specific functions.
- Supplies UI functions common to all (or at least most) forms.
- Contains zero or more views, arranged in some form specific way.

View

- Represents abstract region for displaying some sort of content or UI.
- Not nestable (i.e. a view cannot contain another view)
- Gives forms more flexibility since views are location independent (i.e. all drawing within a view is relative to the view's coordinates)

Preferences

- Manages the storage of persistent preferences used by the application.

EventHandler

- Abstract interface for any component that can handle user events (i.e. Form, View)

Browser Components

Expat

- Collection of open source C files which parse XML files.
- Expat as a shared library to perform parsing, and will subsequently build a parse tree based on its output.

XMLParser

- Wrapper around expat and the parse tree (expat could be replaced by a different parser, and only this class would be affected).
- Manages the construction of the tree, using expat to build a tree composed of XMLElements and XMLAttributes.

XMLElement

- Single XML tag, with attributes and data

XMLAttribute

- Represents a single XML attribute

WMLGlyph

- Graphical representation of XMLElement.

- Based on recursive composition technique in *Design Patterns*, Chapter 1 (by the Gang of Four).
- A glyph can be a container for other glyphs, or can be a leaf.
- Glyphs draw themselves, and handle UI tasks within its boundaries.
- Examples of glyphs are images, blocks of text, and lines of text.

HTTPProtocol

- Implements HTTP Protocol for any underlying session mechanism (i.e. TCP)
- Possibly composed of public domain source code.

PaImOS/WML Browser Glue Components

WMLForm

- Subclass of Form, defines browser specific form features, such as back and forward arrows.
- Contains a WMLView

WMLView

- Displays WML content. Can be embedded in any application that implements the Form and View classes described above.
- Gives WMLGlyph a display area for layout
- Responsible for scrolling WML content

WMLSession

- Responsible for retrieving data, and supplying pages on demand to the WML browser.

- The underlying mechanisms will be pluggable, meaning regardless of whether a modem is used, or a Bluetooth module communicating to a cell phone, the rest of the browser will work in the same way.

TCPSession

- Subclass of WMLSession which uses built-in TCP stack
- Provides socket level instructions to enable HTTP

Generic Classes

Vector

- Easy to use data structure for managing groups of data

Iterator

- Iterator for vector, or any other class which represents a collection of objects

Types

- Defines commonly used types such as Point and Rectangle

Component Interface Design

Application

- main: entry point for application. Calls appropriate function depending on the launch code passed in. Performs suitability checks on environment.
- eventLoop: dispatches commands to appropriate eventHandlers, which have registered themselves. Keeps track of current handler.
- registerHandler: called by forms that can handle events, to tell the application that they should be used with forms of certain ID's. This is necessary because

in the PalmOS, forms are labeled with ID numbers. By registering themselves, form objects become associated with the notion of a form.

- launchNormal: virtual function called by main when program is launched normally. Can be subclassed by applications to customize behavior.

Form

- formLoad, formOpen, formClose: Called by main event loop when this object is the current handler. formLoad should load resources needed to display the form, without writing anything to the screen. formOpen should draw the form itself. formClose should erase the form and release any resources.
- draw: Called by eventLoop (and possibly other functions) when the form's contents should be drawn/redrawn.
- addView. Adds new view to the form.
- handleEvent: Processes form specific events

View

- handleEvent: Processes view specific events
- getBounds Returns the area used by the view

Preferences

- read: reads prefs from storage into temporary data structure
- write: writes prefs from temporary data structure into storage
- Subclasses should provide assessors/modifiers for specific preferences.

EventHandler

- doesHandleType: Returns true if this object handles the specified event type (i.e. menu action)
- doesHandleObject: Returns true if this object handles the specified object (i.e. form with ID #4300)
- handleEvent: Actually handle the event. (Must be overridden by subclass)

Expat

- XML_SetElementHandler: Used to set the callback for new elements. Callback is called for each XML tag seen.
- XML_SetCharacterDataHandler: Used to set callback for data handler.

XMLParser

- parse: parses specified document, returns root element of document (of type XMLElement)

XMLElement

- addAttribute: adds attribute to element
- getAttribute: returns vector of attributes
- getTag: returns tag name

XMLAttribute

- assessor/modifier for name, value pair

WMLGlyph

- intersects: returns true if this glyph intersects specified point

- insert: inserts a sub-glyph into this glyph
- activate: called when the UI element containing the glyph is activated
- deactivate: called when the UI element containing the glyph is deactivated
- draw: recursively draws glyph, and all its sub-glyphs
- getExtent: returns bounds of glyph
- parent: returns parent glyph
- children: returns vector of children glyphs, if any **HTTPProtocol**
- sendRequest: sends request to server, returns requested data. Uses underlying session protocol as provided on device.

WMLForm

- overrides functions of Form, adding appropriate behavior WMLView
- overrides functions of View, adding appropriate behavior WMLSession
- set View: way for view to register itself with a session.
- goTo: called by view to get data of passed in URL

TCPSession

- goTo: overridden from WMLSession
- provides underlying layer for HTTPProtocol to interact with, such as open socket, and read/write data.

Vector

- add: adds element to end of vector
- insert: inserts element into some position in vector

- remove: removes specified element
- removeAll: Empties vector
- getElementAt: returns specified element
- getNumElements: returns count of all elements.

Iterator

- hasNext: returns true if there are more elements
- getNext: returns next element

Data Structure Design

[0022] Important data structures are the representations of the WML document. After parsing, a simple parse tree is created. The parse tree's structure may be based on the notion that a child node is nested within their parent in the corresponding XML. For example, if a run of text looks like:

<p> <i> text </i> </p>

then the element representing 'i' will be a child of the element representing 'p'. Further, the text element will be a child of i. XML attributes are represented as a vector of string pairs, where each pair corresponds to a name and value.

[0023] This parse tree is independent from the glyph tree, although the two trees are similar in structure. The glyph tree has at least one glyph per parse tree element, and possibly more. For example, a block of text that spans multiple lines is represented as a single element in the parse tree. In the glyph tree, however, it will become multiple glyphs — one for each line, and one for the text block itself. This is based on the

Composite pattern, described in *Design Patterns*.

[0024] Vectors are used extensively as a convenient abstraction on top of arrays. STL may be used for data structures such as this. Data structures may be optimized in view of specific constraints faced.

Algorithm Design

[0025] The crux of the browser is the layout algorithm. It should be able to flow a wide variety of objects, and store their position in order to respond to user input (i.e. clicking on a link). It should be flexible enough to work concurrently with data download, so it must operate incrementally. It should be efficient, as computing resources may be limited. To facilitate these needs, the glyph tree to represent on screen information may be used, as described in the previous section. The glyph tree is flexible, in that a block of text can be reformatted into its constituent lines at any point and redrawn, if for example an image is newly downloaded and changes the flow of the document.

[0026] The glyph draw function may be implemented in a relatively coordinate independent way. This means that a glyph may be passed a display area in which to draw, which it can treat as its own canvas. Each glyph may also calculate the sub-areas available to its sub-glyphs. The rendering algorithm is then composed of all glyph types following this general contract. As long as glyphs draw only in the space allotted, and gives their sub-glyphs adequate screen real estate, then the contract is fulfilled. Because of the recursive nature of the algorithm, considerable information can be cached. For example, the extent of a glyph can be stored, so the getExtent function will not need to

examine all sub-glyphs to recompute the extent.

[0027] While the invention has been described with respect to a limited number of embodiments, those skilled in the art, having benefit of the above description, will appreciate that other embodiments may be devised which do not depart from the scope of the present invention as described herein. Accordingly, the scope of the present invention should be limited only by the appended claims.

CLAIMS

What is claimed is:

1. A method for providing multimedia data over an electronic network to a wireless computing device having a browsing program, comprising:
receiving a first portion of multimedia data having a first display format at a proxy server, wherein the first portion has text data and graphical data;
converting the first display format to a second display format, wherein the browsing program interprets multimedia data in the second display format;
and
transmitting the text data over the electronic network to the wireless computing device before transmitting the graphical data and before receiving a second portion of multimedia data.

ABSTRACT

A system and method communicates data from a web content provider to a wireless computing device (e.g., a PDA, a cellular phone) via a proxy server. The proxy server transforms web content “on-the-fly” into a streamlined format, optimizes it for display, and delivers it to the device, where it is progressively rendered by a browser installed on the device.